PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H04N 7/24, 7/26

A2

(11) International Publication Number: **WO 00/65838**

(43) International Publication Date: 2 November 2000 (02.11.00)

(21) International Application Number: PCT/GB00/01614

(22) International Filing Date: 26 April 2000 (26.04.00)

(30) Priority Data:
9909605.9 26 April 1999 (26.04.99) GB

(71) Applicant *(for all designated States except US)*: TELEMEDIA SYSTEMS LIMITED [GB/GB]; Mount Pleasant House, 2 Mount Pleasant, Huntingdon Road, Cambridge CB3 0RN (GB).

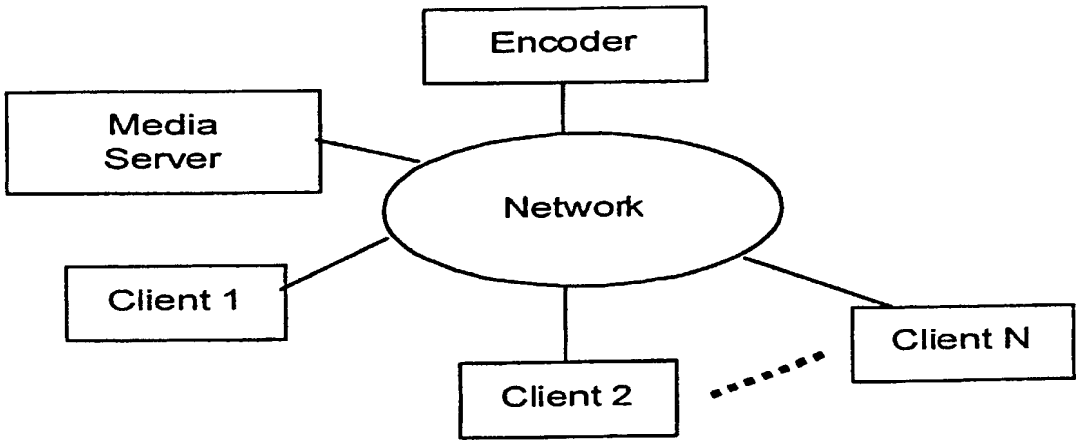
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: KING, Tony, Richard [GB/GB]; 28 Marlowe Road, Newnham, Cambridge CB3 9JW (GB). GLAUERT, Timothy, Holroyd [GB/GB]; 44 Whittlesford Road, Little Shelford, Cambridge CB2 5EW (GB). WILSON, Ian, David [GB/GB]; 26 The Limes, Harston, Cambridge CB2 5QT (GB). EL WELL, Phil [GB/GB]; 4 William Smith Close, Cambridge CB1 3QF (GB).

(74) Agent: ORIGIN LIMITED; 24 Kings Avenue, London N10 1PB (GB).

(81) Designated States: GB, JP, US, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: CONVERSION OF A MEDIA FILE INTO A SCALABLE FORMAT FOR PROGRESSIVE TRANSMISSION



(57) Abstract

The delivery of a single encoded video or audio file which can be played back over a network to various clients at different data rates, resolutions and quality levels, as individually determined by each client, is disclosed. An encoder inserts into the media file bitstreams 'layer signalling' information which delimits and identifies a number of different layers (for example, different 'significance/scale layers' and different 'region layers'). A media Server stores the media files, and can distribute to different networked clients bitstreams with different properties depending upon the layers which are requested by or are appropriate to each client.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

CONVERSION OF A MEDIA FILE INTO A SCALABLE FORMAT FOR PROGRESSIVE TRANSMISSION

Technical Field

This invention relates to methods for the delivery of media files, such as video and audio files, from a server to one or more devices using a network, in which the files stored on the server are encoded so that the generation and/or reception of their associated bitstreams can be controlled to yield a file at the device which meets the specific criteria (for example, image quality) of that device. The network may include the Internet and the device may be a client or another server.

Background to the Invention

One of the problems of delivering media over networks is that the encodings traditionally used to represent media take no account of the bandwidth limitations of networks, or of the varied and possibly conflicting requirements made upon the media by the shared access of multiple clients. Most media file formats have an "all-or-nothing" approach to the way they are viewed: until the media is completely delivered and fully decoded, the user cannot decide whether it is what they wanted. If not, then there is no option but to discard it completely. This is obviously an inefficient use of resources.

Recent work has tried to improve the usability of media over networks, and reduce inefficient use of network resources, by introducing "progressive" download. One example is found in the Encoder Scalability Option in the Indeo product from Intel Corporation.

In such schemes, the media is encoded as an ordered set of discrete blocks. Each block encodes the entire media in such a way that reception of the first block allows some aspect of the entire media object to be viewed, and each successive block received produces a discrete step in the quality of the representation at the client. The client can exert some control over the use of network resource, and over the quality of the

media delivered, by stopping the delivery when the representation has the desired quality.

The client, however, does not have control over the content of the block and therefore over the characteristics of the progressive improvement. Such decisions are made by the content provider when the media file is compiled. Neither does the client have any control over the ordering of the blocks or over the ordering of data within the blocks.

Thus a strictly sequential approach both to content delivery and to the quality progression is enforced, and the parameters which control these aspects are not available to the client.

Also, the quality improvements available from the system are only available in fixed steps, the number of such steps also being predetermined by the content provider.

Another approach to meeting this need for 'progressive image transmission' is disclosed in US Patent No. 5,880,856 to Microsoft Corporation, which teaches a particular approach to the use of wavelet transforms. In the US 5,880,856 patent, an image is transformed using the wavelet transformation to yield 4 or 5 decomposition levels, with a base decomposition level giving a low resolution image, and increasingly higher decomposition levels giving higher resolution images. The client receives initially only the base decomposition level data, but the low resolution image resulting from the base decomposition level data gradually sharpens up as higher decomposition levels are received and decoded. Sharpening up of the image occurs as a result of 2 factors: first, as all of the 4 sub-bands which form each decomposition level are received and decoded, the image quality level increases slightly. As successive decomposition levels are received and decoded, the image quality increases more significantly. One characteristic of the system taught in the 5,880,856 patent is that the sub-bands and decomposition levels are all transmitted to clients in the same, fixed order, which is relatively inflexible. Whilst this scheme allows the client control over quality, it provides no other parameters that can be controlled.

Further reference may be made to the paper by Beong-Jo Kim, Zixiang Xiong and William Pearlman, *"Very Low Bit-Rate Embedded Coding with 3D Set Partitioning in Hierarchical Trees"*, submitted to IEEE Trans. Circuits & Systems for Video Technology, Special Issue on Image and Video Processing for Emerging Interactive Multimedia Services, Sept. 1998. This paper discloses applying a SPIHT compression scheme to a wavelet-based transform, which yields a bitstream encoding multiple spatial resolutions, with progressive quality ordering within a given spatial resolution, as in the US 5,880,856 patent.

Another system for scalable video transmission is disclosed in US Patent No. 5,768,535 "Software-based encoder for a software-implemented end-to-end scalable video delivery system". Here, the scalable property of a hierarchical image decomposition, similar to the wavelet transform, is used in conjunction with vector quantisation to produce a stream that scales in spatial and temporal resolution. At the receiving end the decoder extracts a stream with the required spatial and temporal properties.

A similar system is described in the JPEG 2000 image coding system draft specification (ISO/IEC CD15444-1 : 1999). This describes a coding method for still pictures using a wavelet transform in conjunction with coefficient significance-ordering in a manner similar to the SPIHT system described above. Markers are inserted into the bitstream to allow the decoder to extract a reconstructed image of the desired quality.

Yet another approach is disclosed in US Patent No. 5,953,506 "Method and apparatus that provides a scalable media delivery system", which describes a system for encoding multimedia data in MPEG format as a base stream and a set of additive streams that can be combined to provide various quality levels to satisfy the requirements of different clients. The invention adapts to changing network bandwidth by adding or removing streams, according to a Client-provided profile, and so increasing or reducing bit-rate. It is designed to satisfy a straightforward streaming

model of media delivery, where a media steam is produced by the Server and consumed by the Decoder.

The five prior art disclosures described above assume no storage at the Client, and make no provision for subsequent Client-side operations on the media such as video previewing using fast-forward and rewind, single-frame stepping forwards and backwards, and freeze-frame. These types of operation require additional knowledge of the structure of the file in order to allow media manipulation tools to determine how to refine previously transmitted and stored material so as to improve its quality in a controllable way. In particular, knowledge is required of the way that a component of the media depends on other components, in order for decoding to be possible. They are also specific to particular media encoding formats, in that the tools that manipulate the media in the quality domain need to know the details of the encoding scheme in order to function. Most importantly, such tools cannot be made to be independent of encoding format, and to work correctly with any media file with any media encoding.

New applications for media delivery over networks (e.g. the Internet) require more flexibility and greater interactivity than is possible with any of the prior art schemes described above. Such an application is that of professional media browsing where the aim is rapidly to find, examine and annotate many different aspects or views of the media. Such a system must be capable of the following:-

- Randomly to access any component of the media, where a component can be a time-ordered sequence of media parts, a media part sampled at a particular time, a spatial area within a media part, a colour within a media part, or any other aspect which can be described.

- To access the above components at a quality, spatial resolution or frequency resolution specified by the client.

- To allow the results of any such access to be refined in such a way as to increase the information content and consequently improve the visual quality, amount of detail, or other aspect, of the material.
- To perform the above operations using format-independent tools.

In order to satisfy the requirements listed above a system is required for structuring media files of any type for the maximum flexibility of access of their constituent components.

Summary of the Invention

In a first aspect of the present invention, there is a method of converting a media bitstream, encoded in one of many possible original compression formats, into a layered file in a new format that can be reconstructed as a media file at a device to meet quality criteria specified by the device, the new format remaining the same irrespective of the original compression format so that the media file can be manipulated without detailed information relating to the original compression format used.

The present invention is therefore based on the insight that it is possible to design a universal new layered file format into which files encoded with many different schemes (e.g. block based schemes such as MPEG, JPEG and DV; subband based schemes such as wavelets) can be converted. Media files conforming to this new format can then be manipulated, e.g. at a client device, without any detailed knowledge of the original compression format. Various quality axes define the layering in the new format: extensive manipulation functionality may then be supported by the new layered file format, allowing complex editing and viewing tasks to be performed. An implementation of this new format is the IPV format from Telemedia Systems Limited of the United Kingdom.

Preferably, the method comprises the step of structuring the media bitstream at an encoder using a protocol which converts the media bitstream into a file with a structure including (i) layers which sit in a dependency hierarchy and (ii) groups of those layers which are related by dependency. Hence, the underlying reason why it is possible to convert many different encoding formats into a single universal format is that all such encoding formats share a two-fold structure: the appreciation of this two-fold structure is a further insight on which the present invention is based. A codebook may be used to store information that maps the properties of a particular encoding into a form conformable to a layered representation to generate a consistent view of any media file so that the media file can be manipulated without detailed information relating to the original compression format used; the codebook preferably exploits the two-fold structure insight.

In the IPV implementation, layer information is encapsulated and labelled to form a 'slice': a group of layers related by dependency is referred to as a 'context'. The slice is a transport mechanism for which the layer information is in effect the payload. The 2 fold structure is represented in MPEG, JPEG, 2-d wavelets and 3-d wavelets as follows in the IPV implementation:

	SLICE	CONTEXT
MPEG	A frame	A GOP
JPEG	Single frame	Single frame
2-d wavelet	Spatial data representing a sub-band at a particular bit significance	Single frame
3-d wavelet	Spatial and temporal data	2^n frames

	representing a sub-band at a particular bit significance	

- In the IPV implementation, a media encoding scheme can be described in terms of units of information which represent an input signal captured within a temporal window and digitally sampled, then transformed by a particular coding scheme into a set of conceptual layers These layers initialise or refine particular quality axes within the encoded media file and, again conceptually, can be added or subtracted in order to improve or reduce the quality of the decoded media file. Depending on the encoding scheme used, these layers may operate spatially, temporally, or along any other quality axis. The number of quality axes will vary between encoding schemes, as will the granularity with which quality is represented along each axis. Dependencies will exist between the layers such that it is impossible to decode a layer without also being able to decode the layer on which it depends.

- For a given encoding scheme this implicit or conceptual layering can be made concrete by applying a protocol in a particular way appropriate to that scheme, that effectively breaks up the original file or bitstream along these quality axes.

- Such a protocol can be used to provide a consistent view of media files from the point of view of tools designed for manipulation of these layers. This allows operations to be defined that are valid for any files in this layered format, irrespective of their original encoding.

- An important class of such operations is the creation and use of filters that discard or retain layers so as to meet a particular requirement for quality of the decoded media file. Again, these filters will work correctly irrespective of the original encoding format of the media.

In IPV, we have in essence a method of structuring a media file into ***Layers*** where a Layer carries information that can be used to initialise or refine a particular aspect of the media file. The method defines rules used to split up a particular encoding and specifies the syntax and semantics of the information that is added to the encoding to achieve the layering. The method defines information that must be available globally to tools that manipulate the layered bitstream and the mechanisms by which means these tools are expected to work. Finally, the method identifies, but does not describe in detail, the major components that are required to implement a system: An ***Encoder*** encodes the media, adds the layering protocol, and transmits the media as a bitstream onto a network; a ***Server*** stores the layered bitstream as a network resource; a ***Filter*** selects data for a particular layer or set of layers and appends a ***Filter Mask*** to the bitstream to signal the bitstream content to downstream tools; *a* ***Client*** decodes and views the media.

Here, the term network should be expansively construed to cover any kind of data connection between 2 or more devices. In IPV, the network includes the Internet. The device at which the media file is reconstructed may be a client or may be a server, such as an edge server. A file is any consistent set of data, so that a media file is a consistent set of data representing one or more media samples, such as frames of video or audio levels.

Another aspect of the invention includes a media file which has been reconstructed from a bitstream using any of the above inventive methods.

A further aspect is a computer readable data signal in a transmission medium which can be reconstructed as a media file, in which the data signal is generated using any of the above inventive methods.

A further aspect is a computer program which when running on a client enables the client to receive and playback a media file reconstructed from a bitstream which has been converted using any of the above inventive methods.

A final aspect is a computer program which when running on a server or encoder enables the server or encoder to perform any of the above inventive methods.

Brief Description of the Drawings

The invention will be described with reference to the accompanying drawings, in which:

Figure 1 is a schematic representation of a network used in performing the method of media file delivery according to the present invention;

Figure 2 is a schematic representing the sub-bands which result from applying wavelet transforms to an image in a conventional multi-scale decomposition with examples of partial reconstruction according to the present invention;

Figure 3 is a schematic representation of the format of the 'chunk' data structure according to the present invention;

Figure 4 is a schematic representing the typical path of a unit of media through the layering system according to the present invention;

Figure 5 is a schematic representation of the labelling mechanism as applied to a wavelet encoding according to the present invention;

Figure 6 is an example of a fragment of a Codebook for the labelling example of Figure 5, according to the present invention;

Figure 7 is a schematic representation of slice filtering as applied to the labelling example of Figure 5 according to the present invention;

Figure 8 is a schematic representation of slice merging as applied to the labelling example of Figure 5 according to the present invention;

Figure 9 is a schematic representation of the labelling mechanism as applied to MPEG encoding according to the present invention;

Figure 10 is an example of a fragment of a Codebook for the labelling example of Figure 9, according to the present invention;

Figure 11 is a schematic representation of slice filtering as applied to the labelling example of Figure 9 according to the present invention.

Figure 12 is a schematic representation of the labelling mechanism as applied to DV encoding according to the present invention;

Figure 13 is an example of a fragment of a Codebook for the labelling example of Figure 12, according to the present invention;

Figure 14 is a schematic representation of slice filtering as applied to the labelling example of Figure 12 according to the present invention.

Detailed Description

A. Key Concepts

Block-based, motion compensated encoding schemes.

There are several examples of block-based encoding schemes that use Discrete Cosine Transform, motion detection and compensation to compress video by removing spatial and temporal redundancy from the source. Of these, the most familiar is MPEG (i.e. MPEG-1 or MPEG-2). MPEG utilises three types of compressed picture: I-frames, P-frames and B-frames. An I-frame is built up from 16 x 16 pixel square blocks (Macroblocks) of image, represented as a set of spatial frequencies obtained through the Discrete Cosine Transform (DCT), where the low spatial

frequencies are generally represented with much greater accuracy than the high spatial frequencies. Temporal redundancy is removed using P (predicted) and B (Bidirectional) frames. A particular Macroblock in a P-frame is generated at the encoder by searching a previous I-encoded frame (or P-frame) for a Macroblock which best matches that under consideration. When one is found the vector is calculated which represents the offset between the two. This motion vector, together with the error between the predicted block and the actual block, is all that need be sent in order to reconstruct the P-frame at the receiver. The third kind is called a B (Bi-directional) frame. B-frames are based on motion vectors obtained by past and future I and P-frames; they provide a smoothing effect, increase the compression factor and reduce noise.

Because P-frames are computed from previous P-frames, errors can build up, so it is necessary periodically to insert I frames. A typical MPEG sequence of frames may look like this:-

I B B P B B P B B I B B P B B P B B I.....

From the point of view of the present invention, MPEG exemplifies three properties of encoded media files that are factors in the design of a system such as is described here.

The first property is that the initial sample structure is preserved through the encoding, i.e., the identity of individual frames is not lost in the encoded bitstream. This means that temporal properties can be manipulated by adding or removing frames in the compressed domain. Secondly, a temporal window is defined (the MPEG Group of Pictures or GOP) within which temporal redundancy is exploited to achieve compression. Thirdly, a complex set of dependencies is defined by the encoding; in MPEG, P-frames require I-frames, and B-frames require I and P-frames, for decoding.

There are other examples of block-based encoding schemes that utilise motion detection and compensation including H.261 and H.263.

Block-based intra-frame only encoding schemes.

There are other schemes, notably JPEG and DV (as defined in SMPTE 314M-1999), that use block-based encoding without motion compensation. In both JPEG and DV, the basic scheme is to transform blocks of pixels into frequency components using the DCT, quantise the components by multiplying by a set of weighting factors, then variable-length code the result to produce the encoded bitstream. DV, however, introduces the concept of *feed-forward* quantisation that optimises the compression process prior to compression being applied. To do this the DCT-transformed image is examined and classified into areas of low, medium and high spatial detail. Using this information, different tables of quantisation factors are selected and used according to area, with the object of matching the fidelity with which frequency coefficients are represented, to the frequency response of the human visual system.

A second feature of DV is it's use of block-based adaptive field/frame processing. What this means is that a 64-entry DCT block can represent either an 8-by-8 area of pixels in a de-interlaced frame (an 8-8 DCT), or two 4-by-8 areas in the first and second fields of a frame (a 2-4-8 DCT). The choice between the two is done by detecting motion. The former scheme is used if there is little motion occurring between fields, the latter if motion is detected; this choice being made on a per-block basis.

As with MPEG encoding, three observations can be made about the nature of the encoded bitstream. As before, the sample structure is preserved through the encoding; secondly, a temporal window is defined which, in this case, represents the two fields of a frame. Thirdly, a set of dependencies is defined: for example, dependencies exist between the fields within a frame wherever 2-4-8 DCT blocks have been generated.

Subband encoding schemes.

A more recent alternative to a block-based encoding scheme using a transform such as DCT, is a subband encoding whereby a complete image (rather than small blocks

thereof) is processed into a set of frequency/space limited bands, sometimes referred to as ***scales***. An example of this is the Wavelet Transform.

The wavelet transform has only relatively recently matured as a tool for image analysis and compression. Reference may for example be made to Mallat, Stephane G. "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation" IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.11, No.7, pp 674-692 (Jul 1989) in which the Fast Wavelet Transform (FWT) is described. The FWT generates a hierarchy of power-of-two images or ***subbands*** where at each step the spatial sampling frequency - the 'fineness' of detail which is represented - is reduced by a factor of two in x and y. This procedure decorrelates the image samples with the result that most of the energy is compacted into a small number of high-magnitude coefficients within a subband, the rest being mainly zero or low-value, offering considerable opportunity for compression.

Each subband describes the image in terms of a particular combination of spatial/frequency components. This is illustrated in Figure 2A. Here, a wavelet filter is applied twice to an image. After the first application 4 subbands at Level 0 result, with each subband a quarter the scale of the original; after the second application, four new one-eigth-scale subbands are created. To reconstruct an image fully, the procedure is followed in reverse order using an inverse wavelet filter. The Figure also illustrates the scheme used to name subbands: 'L' and 'H' refer to Low-pass and High-pass wavelet filters which are applied in x and y to generate a subband. Applying the H filter in both x and y gives the 'HH' subband, applying 'L' in x and 'H' in y results in a 'LH', and so forth.

Subbands need not always be fully reconstructed into the original image; they may be combined in different ways to produce a final image to satisfy some set of individual requirements. Three examples are shown in Figure 2. In Figure 2B, the LL sub-band of Level 1 is used as a one-sixteenth scale version of the original. In Figure 2C, the four subbands of Level 0 are inversely transformed to reconstruct a one-quarter resolution

version of the original. In the third example, Figure 2D, the LL and LH sub-bands of Level 1 are used, together with the LH sub-band of Level 0 to reconstruct an original resolution image, but with horizontal features at all scales emphasised.

Image Tree-based Compression.

Tree-based data structures can be used by compression schemes to exploit spatial redundancy in images. A recent development of such a scheme is the SPIHT algorithm; see Beong-Jo Kim, Zixiang Xiong and William Pearlman, *"Very Low Bit-Rate Embedded Coding with 3D Set Partitioning in Hierarchical Trees"*, submitted to IEEE Trans. Circuits & Systems for Video Technology, Special Issue on Image and Video Processing for Emerging Interactive Multimedia Services, Sept. 1998.

The SPIHT algorithm is an effective compression step for use in conjunction with a wavelet transform, since it can efficiently exploit the decorrelation of the input image provided by the transform to get high levels of data reduction. For the purposes of the present invention a key feature of the SPIHT algorithm is its ability to analyse the transformed image in terms of ***Significance***. It efficiently locates and partially orders all the samples with respect to bit-significance, i.e., the position at which the highest-order bit in a sample is set. Since this corresponds to the magnitude of the sample, the samples are effectively ordered with respect to their energies, or the contribution they make to the reconstructed object. A significance layer is generated by choosing a bit-position and outputting the value of that bit-position (1 or 0) for all the samples for which that bit-position is defined.

As with MPEG encoding, a set of dependencies between related image components is defined by schemes such as Wavelet/SPIHT compression. In this case, a significance layer can be decoded only if the next highest significance layer can also be decoded. Similarly, to decode a subband (e.g. LH0 in Figure 1) the parent subband (LH1 in Figure 1) must also be decoded.

3D Schemes

The Wavelet and SPIHT compression schemes can be extended to the third (time) dimension in a fairly straightforward manner. In this case a sequence of frames are captured and are treated as a 3-dimensional block of data. The wavelet filter is applied along the time dimension as many times as it is along the vertical and horizontal dimensions, resulting in a transformed block comprising a set of spatio-temporal cubes, each of which is a subband. The 3D-SPIHT algorithm represent these subbands in terms of octtrees (as opposed to quadtrees in the 2D case) and generates the bitstream accordingly.

Other compression schemes can also be used in the present invention, as will be appreciated by the skilled implementer.

B. System description

The system capable of performing the method of the present invention comprises an Encoder, a Network, a Server and N clients, as shown in Figure 1.

A Encoder compresses the incoming media, structures it into layers and adds layer labelling information as described in the next section, prior to transmitting it as a bitstream over a digital communications network, to be stored on the Server.

In Figure 1, there are N clients, each of which engages in a session with the media Server during which media content and control information are transmitted. For each client a control channel to the Server is maintained which allows that client to request that media be transmitted to it. The types of requests that can be made include (but are not limited to) the following:-

- Seek to a particular point in the media file.
- Transmit a media clip at a particular quality.
- Transmit extra information to improve in a specified way the quality of the media at the Client.

These requests support a variety of useful application-level functions at the Client such as video previewing using fast-forward and rewind, single-frame stepping forwards and backwards, and freeze-frame. For example, in a media browsing application, a low-quality version of the media may be sent to the client with very little latency, to allow the user to start work immediately, for example, on rapidly scanning through a file to locate items of interest. Initially the user may not so much be interested in the absolute quality of the media, as in the responsiveness of the system to requests to move rapidly from one point to another. Having found an item of interest the application may then determine that this particular section of the media is to be rendered at greater quality and issue an appropriate request to the Server. The Server calculates which extra layers should be delivered to improve the media information in the way required and transmits only those layers. Note that no extra processing of the media file is required during this procedure other than to locate and send the appropriate sets of data. Note also that the quality improvement may occur in a number of ways, including:

- Increasing the temporal resolution of the media (i.e., for video, allow more individual frames to be resolved).
- Increasing the spatial resolution of the video.
- Increasing the sampling frequency of the audio.
- Decreasing the degree of distortion of the media.

Central to the working of this scheme is the mechanism used to convert a media file in any encoding format, including those described in the preceding section, into a layered format with properties that support the operations described above. This is the purpose of ***Layer Labelling*** as described in the next section.

C. Description of Layer Labelling

Overview

A Layered bitstream is built out of a media encoding by splitting it up into packets (called ***chunks*** here) in a manner determined by the encoding type, and appending ***Labels*** to the chunks that uniquely describe their contribution to the reconstructed file. In order that distributed tools can be built and configured to understand the formats of many layered bitstreams in a distributed environment, ***static*** signalling information (constant throughout a stream's lifetime) is made available, either as a reference to a globally available file, or as data sent once in the bitstream. For configuration information that must vary within the stream, provision is made for ***dynamic*** signalling information to be carried in the bitstream.

To be successful the labelling scheme must do three things well:

- Efficiently and controllably allow parts of a media file to be selected to obtain a particular quality of service to satisfy media quality, bit-rate or latency requirements.
- Efficiently support the location and transmission of those parts of the media necessary to reconstruct a particular item to a specified quality.
- Encapsulate the underlying media encoding and hide complex details while offering a uniform, media encoding-independent, layered view of the file.

The first requirement is addressed through the concept of a ***Filter*** that specifies which parts of the file are to be selected for transmission. The second requirement needs in-band ***signalling*** information to describe and delineate the layered structure within the bitstream. The third requirement is addressed through an indirection mechanism that maps the subtleties of a particular encoding to the uniform layered view. This indirection mechanism is termed the ***Codebook***. A Codebook is defined for every new encoding scheme, or variations of existing schemes with new parameters.

Layer Labelling Format

This section defines the format of the layer labels used in the present invention to convert a media file to a layered stream format. Streams are stored and transmitted as contiguous groups of chunks, each with a chunk header defining the length and type of the chunk. Some chunk types are reserved for carrying data (termed slices), whereas others are used for signalling.

Extra configuration information for these streams, such as encoding format and slice dependencies, can be associated with the data stream. The static portion of this information can be stored externally, or in-band using signalling chunks; dynamic information is always stored in-band.

Chunk Header

All chunks are introduced by a chunk header, the format of which is illustrated in Figure 3. All chunks, irrespective of their type, are introduced by one of these headers. The result is a data stream whose structure is fixed, while still allowing room for expansion in the future.

Type

The 2-bit type field is used to distinguish between 4 independent sets of chunk types. (The issue of independence is important, since there is no way to specify a dependency between two labels belonging to different types.) The following type values are currently defined:

data chunks (slices)

signalling chunks

Length

The 22-bit length field gives the length of the chunk in bytes, including the header. A chunk containing no payload information (i.e. just a header) would, therefore, have a length of 4.

Label

The 8-bit label field is the layer label for the chunk. The interpretation of the label is dependent on the type field (see above), so the header format allows for 4 sets of 256 labels.

Terminology

Data chunks (equivalently termed *slices*) are used to encapsulate a stream of multimedia data. The labels attached to the slices are used to indicate the nature of the slice contents and their dependency relationships with other slices in the stream. Signalling chunks can be used within the stream to pass extra information about the data, such as details about how it has been encoded or filtered.

Slice

The manner in which the slice label allocation is done depends on the exact nature of the multimedia data being encapsulated. There are a set of basic schemes which are mutually agreed between all the tools, but new allocation schemes can be added in order to tailor the system to suit different situations. The only restrictions are:

- slices within a data stream must all be labelled using the same allocation scheme;
- slices must be labelled uniquely with respect to the data they contain; and
- label allocation must be done in slice dependency order.

The first restriction is self explanatory, but the others merit further discussion. The slice label is taken as an indication of the data contents of the slice, and is used by the system to separate and merge the slices to obtain data streams of differing quality. In order to allow the stream decoders to be stateless, the slice label must identify uniquely the actual data contents. As a result, slices which are too long to be encapsulated as a single chunk must be split in such a way as to enable the decoder to determine which is the base data and which is the continuation information.

The slice labelling must also reflect the dependency hierarchy, with all slices having numerically lower label values than those which depend on them. This means that when the slices of a stream are encoded in dependency order, the label values increase in line with the progress through a dependency unit (see "context" below). Not only does this simplify the job of merging the slices from data streams which have been split previously, but also allows the boundaries between contexts to be implicit (if a slice label is not numerically greater than its predecessor, then it must belong to a new context).

Context

A group of slices which are related by dependency (either spatial or temporal) is called a "context". As described above, the label values for the slices within the context must be arranged so that they are in numerically increasing order. This usually allows the boundaries between contexts to be detected implicitly, without the need for an explicit signalling chunk. Based on the label allocation scheme described above, slices within the context will also, therefore, be arranged in dependency order such that slices always precede those which are dependent on them.

Because of the number of slice labels available (and the restriction of unique slice labelling), it follows that the maximum number of slices which comprise a context is 256. The dependency hierarchy for the slice labels is defined in a code book (see below) and is fixed for the duration of the data stream. Note that it is not mandatory for slice labels to be allocated consecutively (i.e. gaps in the allocation scheme are allowed); nor is it essential for a specific slice label to be present in all contexts.

As well as grouping together slices which are related by dependency, contexts have a very important rôle in defining the boundaries at which editing can be performed. There is no need for explicit signalling chunks to determine valid edit points.

Contexts are assumed to be separated from their neighbours in the temporal domain, and be independent of them; in other words, when contexts are extracted from the data stream, each should be capable of being decoded as a stand alone entity. The one exception to this is exemplified by encoding schemes such as MPEG which have temporal dependencies between adjacent Groups Of Pictures. This is handled by associating an "overlap" with each context allowing temporal units (see sample below) to be marked as dependent on the immediately preceding context.

The overlap value, stored in the media header (see below), defines how these inter-context dependencies are handled: an overlap value of n implies that the first n samples of a context are dependent on the previous context. Since the normal situation is for contexts to be temporally self-contained, the default overlap value is zero. In the case where it is not zero, the slice dependency hierarchy must reflect the inter-context peculiarities of the encoding scheme, with "phantom" dependencies being added if necessary.

Sample

Multimedia data streams are usually represented as a sequence of discrete units, each with a well defined temporal position in the stream. This is particularly true with video data, which is usually modelled as a set of separate frames (even though the frames may have been reordered and grouped in the encoding process, such as with MPEG). The data streams preserve the natural temporal unit of the encoding scheme, with each discrete unit being termed a sample. Whether a context contains a single sample or a group of samples is dependent on the encoding technique used, but specific techniques usually follow a very rigid repeating pattern. By default, therefore, each context is assumed to contain a fixed number of samples, with the context spacing (samples per context repeat count) defined in the system header (see below).

The distinction between contexts and samples is important when contemplating the temporal dependencies of the multimedia data (such as a 3-D Wavelet/SPIHT scheme) and the ability to perform temporal decimation (such as playing every fourth frame of a video sequence). A context contains the smallest number of samples which make up a temporal encoding unit (GOP for MPEG, GOF for Wavelet/SPIHT), with the spatial and temporal dependencies being handled in exactly the same manner. Within the context, each slice is given a temporal priority (see below) which allows for intelligent decimation of the temporal sequence, but does not in itself imply any kind of temporal dependency.

System Header

The system header is used to define the stream attributes which are relevant at the system layer only. Basically, this means parameters which are needed by the Server to provide the level of service required of them. For example, the default context spacing is needed in order to construct the mappings between contexts and samples.

Media Header

The media header is used to define the stream attributes which are relevant at the media layer only. This means parameters which are needed by the stream decoder in order to make sense of the data stream, but which are not required by the Server. Examples which fall into this category are: the horizontal and vertical resolutions of a video encoding; the inter-context dependency overlap; and a reference to the code book which has been used for the encoding.

The reason there is a separate media header chunk, rather than combining the information with that in the code book is because code books are generic in nature and tend to be independent of the raw properties of the original media. The media header fills in these details for a specific recording, thus greatly reducing the number of code books which are required by the system as a whole.

Code Book

The code book is used to define the dependency and quality relationships of the slices within a context, and is necessary in order to construct a filter for a data stream. Irrespective of whether the dependencies are spatial or temporal, the relationships between the individual slices can be represented as a simple hierarchy: each slice type (i.e. slice with a specific label) has a predetermined set of other slice types on which it is dependent. The nature of the hierarchy is, obviously, dependent on the labelling scheme used; but the code book technique allows a great deal of flexibility providing the basic rules of slice labelling are not violated (see above).

In the context of the code book, "dependency" has the following meaning. For one slice to be dependent on another, it must be impossible to decode the slice without it. (This is true, for example, with MPEG video where the I frame is necessary before the first P frame can be decoded.) Note that this dependency relationship says nothing about the desirability of slices when it comes to producing results of acceptable quality.

As well as defining the dependency hierarchy, it is the job of the code book to store a mapping between the individual slice values and the "quality" of the decoded data stream which include those slices. The stream quality is, by necessity, a vague and subjective factor compared to the strict slice dependencies described above; indeed it is likely that there is more than one quality dimension. For example, 3-D Wavelet/SPIHT encoded video can be thought of as having four quality axes: scale (image size), fidelity (image quality), colour and temporal blur.

The bulk of the code book comprises a series of tables (one for each slice label) containing the dependency data, the temporal priority and a set of quality parameters, one for each quality axis. The number and type of quality axes defined is dependent on the nature of the multimedia data and the encoding scheme used. To simplify matters, three important restrictions are applied to quality axes:

Quality axes are assigned names (or quality tags) which have global significance. Code books which employ the same quality tag must use the same allocation scheme when it comes to quality parameters for the relevant axis.

Quality parameters are always specified with regard to the highest quality (unfiltered) version of the data stream. In the case of video, for example, this allows a common code book to be used irrespective of the image size.

The quality parameters themselves must, where practical, be independent of the actual encoding scheme used. For example, the scale parameters for video data might represent the width or height of a scaled image compared to the original, assuming an unchanged aspect ratio.

The code book header contains the list of quality tags for which quality parameters will be found in the tables which follow. This, combined with the restrictions outlined above, allows for filter creation to be done in a manner which is entirely independent of the multimedia data type and encoding scheme. A filter created using a quality axis which is either absent or unspecified always results in slices being unfiltered with respect to that axis.

Temporal Priority

It is often necessary to decimate a data stream, either to reduce the bandwidth requirements or to play it at faster than normal rate. Given the different data encoding schemes, it is not always sensible to work with naïve sub-sampling schemes such as "one sample out of every four". There are two reasons for this.

The inter-slice dependencies of the relevant samples may prohibit simple decimation. This is true, for example, with MPEG video, where there is a complex web of temporal dependencies between the I, P and B frames.

When attempting to maintain a cache of slice data and ensure effective use of available bandwidth, it is essential that repeated (or refined) sub-sampling requests should be carefully orchestrated so as to maximise the data overlap.

Accordingly, each slice in a context is assigned a temporal priority which is used to control decimation. For example, in MPEG video, slices belonging to the I, P and B frames would be allocated temporal priorities of 0, 1 and 2 respectively. It is these temporal priorities which are used when performing temporal filtering below the level of the context.

Signalling Chunks

The signalling chunks are used to identify and annotate the data chunks in a data stream. Whether they are stored together by Servers or generated on the fly at the time of delivery is an implementation detail which is outside the scope of this document. The signalling chunks fall into three distinct categories: static, dynamic and padding.

Static chunks

The static chunks define parameters which do not change during the lifetime of the data stream. This includes the definitions of the basic default values which apply to the whole stream, such as the identity of the code book which has been employed to generate the data slices. Since these chunks are static, there is no necessity for them to be included as part of the data stream: it is just as valid for the information to be sent out-of-band or, as in the case of re-usable code book tables, by reference. If they are transmitted or stored in-band, however, they must be unique, precede the first data slice and be stored in numerically increasing opcode order.

Dynamic chunks

The dynamic chunks define those parameters which vary depending on their position within the data stream. This includes filtration information, since the filters used to generate the stream can vary as the slices are delivered. It also includes all the variable context and sample information, such as the indication of "seek" operations and the handing of contexts with a context spacing which is not constant. Dynamic chunks, by their very nature, carry positional information within the data stream, and hence

must always be sent in-band. Where present, dynamic chunks are only valid at context boundaries.

Padding chunks

The padding chunks have no semantic influence on the data stream, and hence can be positioned at any chunk boundary.

Static Signalling Chunks

System Header

Type: Static

Opcode: 0x00

Contents: Information which is specific to the system layer only (Media Server), and is not required to filter or decode the data stream. The parameters are:

context spacing

Format: Name/Value pairs.

Status: Mandatory.

Position: Out-of-band, or in-band before first data slice.

Media Header

Type: Static

Opcode: 0x01

Contents: Information which is specific to the media layer only (Decoder), and is not required for the operation of the Media Server. The parameters are:

original media parameters

context dependency overlap

code book reference

Format: Name/Value pairs.

Status: Mandatory.

Position: Out-of-band, or in-band before first data slice.

Code Book

Type: Static

Opcode: 0x02

Contents: Information which is specific to the generation of slice filters only, is not required for the operation of the Media Server or Decoder.

quality tag list

slice dependency information

slice quality information

slice temporal priority information

Status: Mandatory.

Position: Out-of-band, by reference, or in-band before first data slice.

Meta Data

Type: Static

Opcode: 0x04

Contents: Information which is pertinent to the data stream, but which is not required for its storage, filtration or transmission. Examples are:

date, time & location

edit history

copyright notice

Format: Name/Value pairs.

Status: Optional.

Position: Out-of-band, or in-band before first data slice.

User Data (1,2,3,4)

Type: Static

Opcode: 0x05, 0x06, 0x07, 0x08

Contents: Private information, added into the data stream by the generator of the encoding, but which is not interpreted by the system in any way.

Format: Opaque byte array.

Status: Optional.

Position: Out-of-band, or in-band before first data slice..

Dynamic Signalling Chunks

Baseline

Type: Dynamic

Opcode: 0x80

Contents: Information regarding the stream's current position with respect to the context and sample sequence numbers, and is usually used to indicate a "seek" operation within the data stream. Has two parameters, either of which can be omitted if the value can be inferred from the stream position:

context sequence

sample sequence

At the start of a data stream, unless otherwise specified, it is assumed that the context and sample sequence numbers both start at zero.

Format: 64-bit little-endian binary.

Status: Optional.

Position: In-band at context boundaries.

Context Start

Type: Dynamic

Opcode: 0x81

Contents: A *context spacing* for the immediately following context.

Format: 32-bit little-endian binary.

Status: Optional. Only necessary if the *context spacing* is different to the default.

Position: In-band at context boundaries.

Context End

Type: Dynamic

Opcode: 0x82

Contents: None.

Format: n/a

Status: Optional. Only necessary if context boundary cannot be determined by simple comparison of slice type values.

Position: In-band at context boundaries.

Filter Definition

Type: Dynamic

Opcode: 0x83

Contents: Used to encode stream filtration information for the data slices which follow. It takes the form of a pair of bit sequences (possibly compressed) indicating which data has been filtered out of the subsequent stream.

The *slice mask* indicates which of the 256 possible slice types have been filtered out of the data stream. It does not imply that the remaining slice types will be found in the stream; nor does it guarantee that the slices

which are present conform to the dependency criteria specified in the code book.

The *context mask* is used to indicate which whole contexts have been filtered out of the data stream. Each context has a unique position in the sequence, and mask refers to the position value modulo 240 (a number chosen because of its numerical properties, being the common multiple of 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24 and 30). Unlike the slice mask, the context mask can be used to determine which contexts are present and which are absent. In the case of inter-context dependencies, however, there is no guarantee that the relevant information is actually present.

Status: Optional.

Position: In-band at context boundaries

Padding Signalling Chunk

Padding

Type: Dynamic

Opcode: 0xff

Contents: Opaque byte array. Padding chunks may be necessary in the case where fixed bit-rate streams are encoded or specific byte alignments are required. They can also be used to re-label a slice in place for data filtration experiments.

Format: Opaque byte array.

Status: Optional.

Position: In-band.

Layer Labelling, Filtering, Merging and Codebook Examples.

Figure 4 gives an overview of some typical transformations undergone by a media file as it is layered, filtered and decoded. In particular, the Figure shows the two styles of filtering that are possible: filtering at the Slice level (within a dependency unit), and filtering at the Context level (across entire dependency units). Slice-level filtering can be used to change the colour depth, spatial resolution, fidelity or temporal resolution of the media. Context-level filtering has a coarser 'grain' and is designed mainly to support temporal decimation of the media.

D. Wavelet Example

Figure 5 shows a more detailed example of labelling Wavelet/SPIHT-encoded data. A depth 2 wavelet decomposition with four significance levels results in a Context of 28 slices. A fragment of one possible Codebook for this encoding is shown in Figure 6 and an example of how the Codebook is used to generate a Filter is illustrated in Figure 7.

Referring to Figure 6, the header of the Codebook includes a **CodebookID** field for self-identification, a **QualityTags** field that specifies the quality axes available for this encoding, a **QualityDivisions** field that specifies, using plain-text names, how those axes are labelled, and an **OriginalSize** field that specifies the spatial resolution of the original media, to provide a baseline for scale calculations.

Each encoding scheme may have its own individual Codebook defined, but any Codebooks that share common **QualityTags** entries must use exactly the same **QualityDivisions** scheme to label the quality axes. Textual names are used to map properties of the different media encodings, which may be similar but not identical, into a single quality scheme. This results in an important capability of the system: that an application can be written to manipulate media using these names in a manner that is completely independent of encoding format.

In the Figure it is assumed that a file manipulation tool wishes to extract a medium fidelity, monochrome, half-scale version of a 352 by 288 video encoding. To accomplish this the Codebook is searched (in practice this is optimised using pre-calculated indices) for the highest-numbered label with the required value of **QualityParams**, i.e., "scale=half, fidelity=medium". Having found this label (13) all the dependent labels are visited until the dependency graph is completely resolved. As this proceeds the bit mask representing the Slice Filter is built: a 'zero' at bit position *n* indicates that the slice with label (*n*) is required in the filtered bitstream, a 'one' means that it should be absent. The result of using this filter is illustrated in Figure 7.

Filtered slices are preceded by a Filter Definition signalling chunk which holds the slice mask used to perform the filtering. This value can be used by downstream tools that require knowledge of the stream characteristics, for example, to initialise a decoder, or set up memory areas for buffering. The Slice Mask is also used when filtered bitstreams are merged to determine the slice content of the resulting stream, as illustrated in Figure 8. Here, a refinement stream containing slices 16, 17, 20, 21, 24 and 25 is merged with the half-scale, low-resolution filtered stream of Figure 7 to obtain a full-scale, low-resolution encoding. The slice mask for the combined stream is obtained using a simple bitwise OR of the two input Slice Masks.

E. MPEG Example

Figure 9 shows an example of labelling MPEG-encoded data, and Figure 10 shows a fragment of the corresponding Codebook. In this example it is assumed that a file manipulation tool wants to generate a Filter that operates on slices in such a way as to generate a fast playback stream. In this encoding a Slice represents an MPEG frame, so the effect of the Filter is to decimate the stream temporally. From the Codebook the tool resolves the dependence hierarchy for the **QualityParams** playrate=fast and builds the Slice Filter as described above. The result of applying the Filter is shown in Figure 11.

F. DV Example

Figure 12 shows an example of labelling DV-encoded data, and Figure 13 shows a fragment of the corresponding Codebook. In this example the DV-encoded data is labelled according to the following scheme: data that represents both 2-by-4-by-8 and 8-by-8 DCT blocks from field 1 are assigned slice label 0; data that represents 8-by-8 DCT blocks only from field 2, are assigned slice label 1. This results in two layers, a base layer with half-vertical resolution derived from field 1, and the refinement layer that produces the full-resolution reconstruction. Figure 14 illustrates the result of applying a Filter that selects label-0 Slices only, so producing a half vertcal resolution video stream.

G. Appendix 1

Some Definitions of Terms, as used in the context of the illustrated embodiments

Scale

A function can be analysed into a set of components with different time/space/frequency content. These components are called scales, and the process of analysis into scales is called multi-scale decomposition. The analysis is performed by a waveform of limited duration and zero integral, called a wavelet. Components that are highly localised in space or time but have an ill-defined frequency spectrum are small-scale and capture fine detail. Components that are spread through space but have a precisely-defined frequency spectrum are large-scale and capture general trends.

Slice

A slice is a labelled encapsulation of a layer.

Layer

A Layer is a conceptual part of a media file that initialises or refines a single homogeneous component of that file, where that component is data representing spatial resolution, temporal resolution, sample fidelity, colour or any other quality axis.

Base Layer

A layer which is sent first to a client to initialise a representation of a media object at that client, to which refinement layers are added.

Refinement Layer

A Layer which is sent to a client subsequent to a Base Layer and which 'improves' the quality, according to some metric, of the representation of a media object at that client.

Significance Layer

A layer where all the refinement information refers to a particular bit-position for all the coefficients undergoing refinement.

Scale Layer

A layer for which all the refinement information is for a particular scale.

Region Layer

A layer for which all the refinement information is for a particular connected region in a space or time-varying function.

Distortion

The distortion of an image can be measured in terms of it's Peak Signal-to-Noise Ratio (PSNR) where PSNR = 10log (255^2/MSE) dB, and MSE is the image's Mean Squared Error.

QualityTags

The set of axes that are available to represent different aspects of perceived quality, for example, spatial resolution, temporal resolution, fidelity of a reconstructed pixel with respect to the original, etc.

QualityDivisions

The marking scheme for the **QualityTags** axes, for example, the spatial resolution axis may be marked with low, medium, high.

QualityParams

A set of classifiers that can be used to describe the contribution of an item of encoded media to the perceived quality of the reconstruction. A **QualityParams** classifier is defined by a (**QualityTags=QualityDivisions**) pair, for example, fidelity=6, or resolution=high.

Codebook

A table that provides an abstraction mechanism such that quality-manipulation operations can be defined on media files that are valid irrespective of their original compression format. The **Codebook** achieves format-independence through use of the **QualityParams** classification system.

Filter

An information structure that defines a partitioning of a layered media file into two parts; one part representing the media file at a reduced quality, and the other representing the enhancement information needed to reconstruct the original file. The simplest implementation is a bitmask where 'zero' and 'one' at bit positions *n* specifies whether a data item with a particular label (*n*) is or is not required in the lower-quality output file.

Filter Mask

A piece of information that is appended to a filtered file in order to inform downstream tools about the information content of the file (i.e, what has been filtered

out of it). If the filter is implemented as a simple bitmask then the filter mask can simply be a copy of this filter.

5

Claims

1. A method of converting a media bitstream, encoded in one of many possible original compression formats, into a layered file in a new format that can be reconstructed as a media file at a device to meet quality criteria specified by the device, the new format remaining the same irrespective of the original compression format so that the media file can be manipulated without detailed information relating to the original compression format used.

2. The method of Claim 1 comprising the step of structuring the media bitstream at an encoder using a protocol which converts the media bitstream into a file with a structure including (i) layers which sit in a dependency hierarchy and (ii) groups of those layers which are related by dependency.

3. The method of Claim 2 further comprising the step of the encoder adding a unique label to each layer as part of the protocol.

4. The method of any preceding Claim in which the quality criteria include one or more of the following: scale, fidelity, colour, temporal blur.

5. The method of Claim 2 wherein the particular aspects of a media file that a layer can represent include temporal resolution, spatial resolution, sample distortion and colour depth.

6. The method of Claim 3 wherein allocation of labels to layers is done such that all layers have a numerically lower label value than those that depend on them.

7. The method of Claims 1-6 in which the encoded file is generated using a wavelet transform and subsequent compression.

8. The method of Claim 7 in which the subsequent compression is SPIHT compression.

9. The method of Claims 1-6 in which the encoded file is generated using block-based transform and subsequent compression.

10. The method of Claim 9 in which the encoding scheme is DV.

11. The method of Claim 9 in which the subsequent compression utilises motion compensation.

12. The method of Claims 9 and 10 in which the encoding scheme is MPEG.

13. The method of any preceding Claim wherein a codebook is used to store information that maps the properties of a particular encoding into a form conformable to the layered new format.

14. The method of Claim 13 where each codebook entry represents a label.

15. The method of Claim 14 where each codebook entry includes a field that specifies the set of labels upon which the current label depends.

16. The method of Claim 15 where each codebook entry includes a quality parameter field that specifies the quality of reconstructed media that results from the inclusion of a layer with the label associated with that entry.

17. The method of Claim 16 where the quality parameter field specifies qualities along multiple quality axes.

18. The method of Claim 17 where the quality parameter field utilises textual names to specify qualities along a particular quality axis.

19. The method of Claims 13-18 where different codebooks are used to represent different encodings.

20. The method of Claims 13-18 where different codebooks use identical quality axes to represent quality parameters within the media.

21. The method of Claims 13-18 where the textual names that specify qualities along a particular quality axis are allocated and used in an identical way in all codebooks that use that quality axis.

22. The method of Claim 2 wherein slices are selected from, or merged into, a stream so that upon decoding the stream, the media file received by the device satisfies criteria specified by that device.

23. The method of Claim 22 wherein a slice filter is used to define the selection of slices.

24. The method of Claim 23 wherein the slice filter is derived from the dependence hierarchy of slices.

25. The method of Claim 24 wherein a codebook is used to provide the dependence hierarchy for slices for a particular encoding scheme, in order to derive the slice filter.

26. The method of Claim 23 wherein the slice filter is implemented as a bitmask where 'zero' and 'one' at bit positions *n* specifies whether the slice with label (*n*) is or is not required in the filtered bitstream.

27. The method of Claim 2 in which a group of slices related by dependency form a context and one or more contexts are selected from, or merged into, a stream so that upon decoding the stream, the media file received by the device satisfies criteria specified by that device.

28. The method of Claim 27 wherein a context filter is used to define the selection of contexts.

29. The method of Claim 28 wherein the context filter is implemented as a bitmask where 'zero' and 'one' at bit positions *n* specifies whether the context with label (*n*) is or is not required in the filtered bitstream.

30. The method of Claim 29 wherein a bitwise OR operation is used to compute a context filter value resulting from the merging of two filtered streams.

31. A method of converting a media bitstream, encoded in one of many possible original compression formats, into a layered file in a new format that can be reconstructed as a media file at a device to meet quality criteria specified by the device, in which a codebook is used to store information that maps the properties of the original compression format into a form conformable to a layered representation to generate a consistent view of any media file so that the media file can be manipulated without detailed information relating to the original compression format used.

32. The method of Claim 1 comprising the step of structuring the media bitstream at an encoder using a protocol which converts the media bitstream into a file with a structure including (i) layers which sit in a dependency hierarchy and (ii) groups of those layers which are related by dependency, the protocol further including the step of adding unique labels to each slice as encoding proceeds and wherein a codebook defines the dependency hierarchy for each layer label and also defines certain quality relationships of the layers in a group oflayers.

33. The method of Claim 32 in which the codebook references a limited number quality parameters.

34. A media file which has been reconstructed from a bitstream using any of the above methods.

35. A computer readable data signal in a transmission medium which can be reconstructed as a media file, in which the data signal is generated using the method of any preceding claim.

36. A computer program which when running on a client enables the client to receive and playback a media file reconstructed from a bitstream which has been converted using any of the above methods.

37. A computer program which when running on a server or encoder enables the server or encoder to perform any of the above methods.

38. A system for delivering media files over the Internet comprising an encoder for performing any of the methods in Claims 1 – 33; a server for storing encoded layered files; a filter to specify which elements of an encoded layered file stored on the server are to be transmitted; and an output connection to deliver the specified elements to a client device over the Internet.

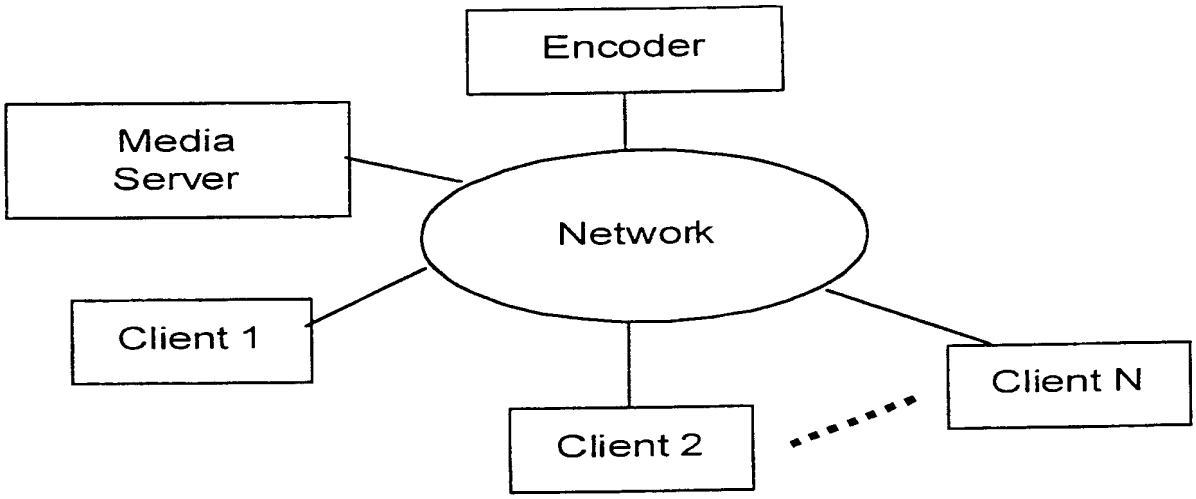
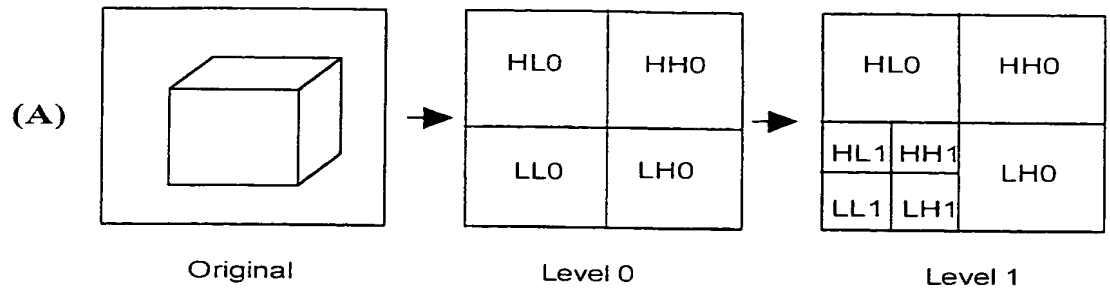
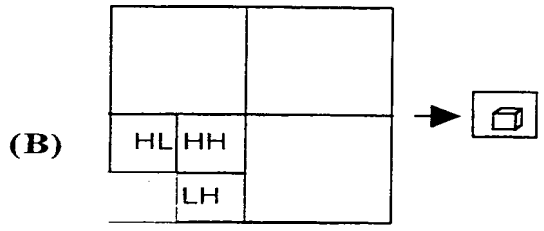
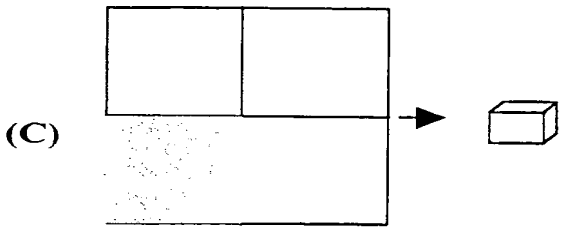


Figure 1







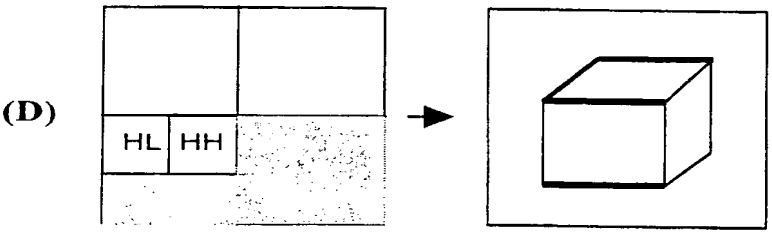


Figure 2

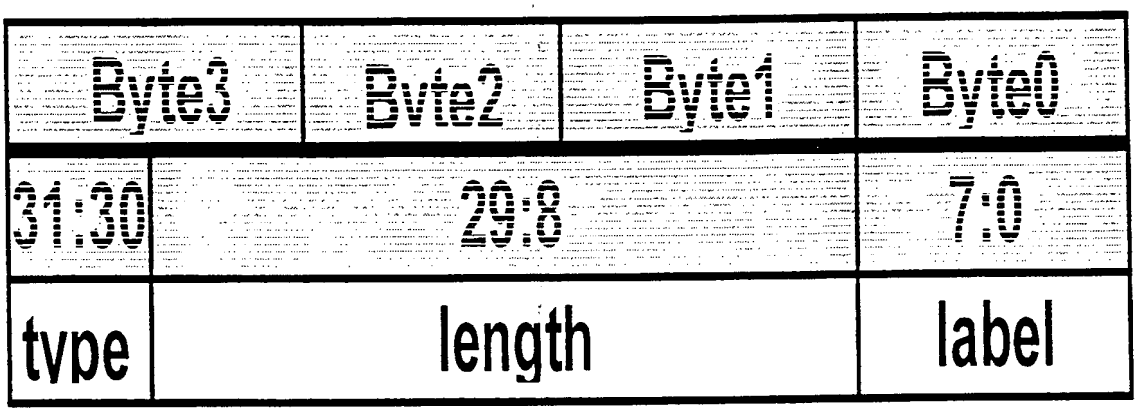


Figure 3

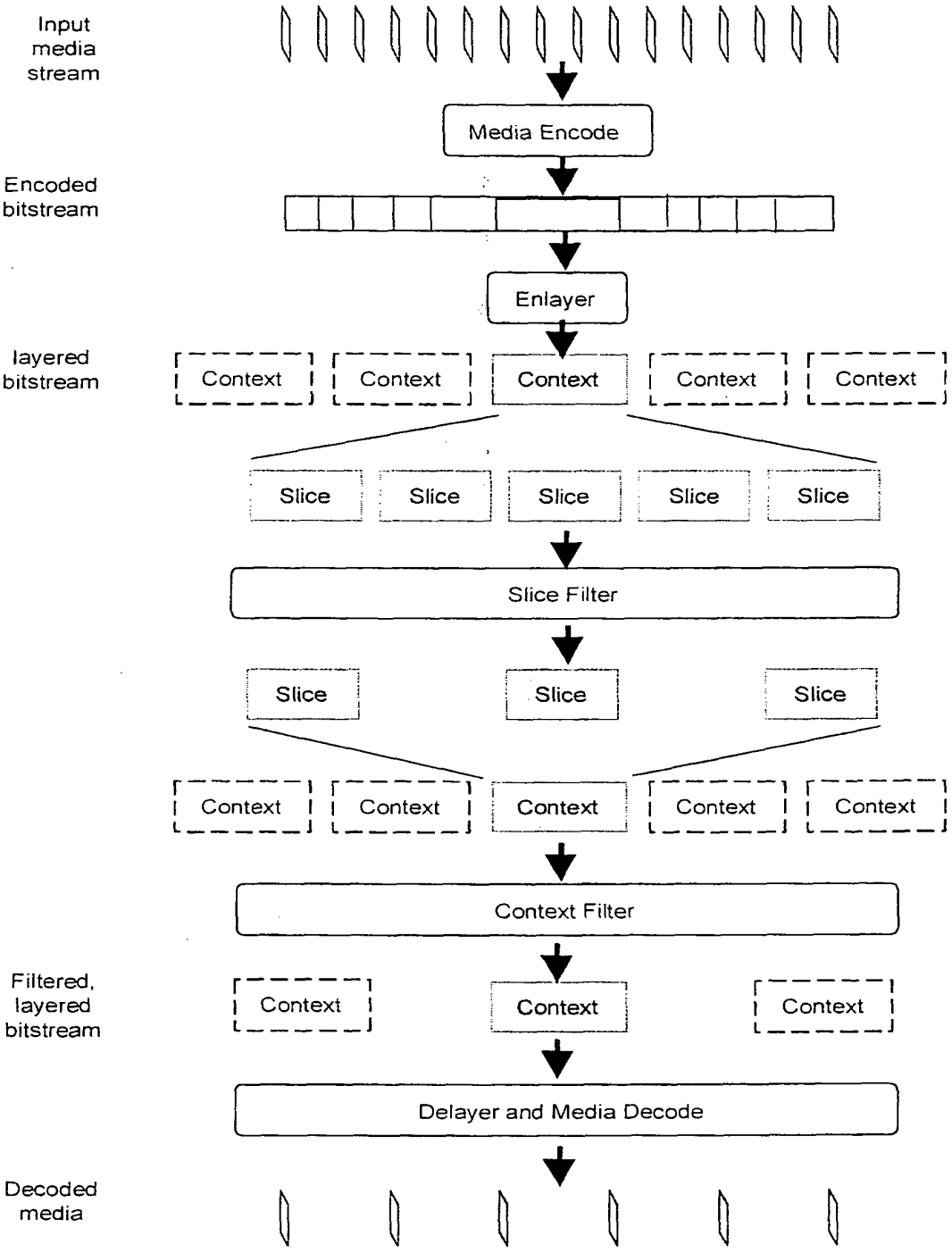


Figure 4

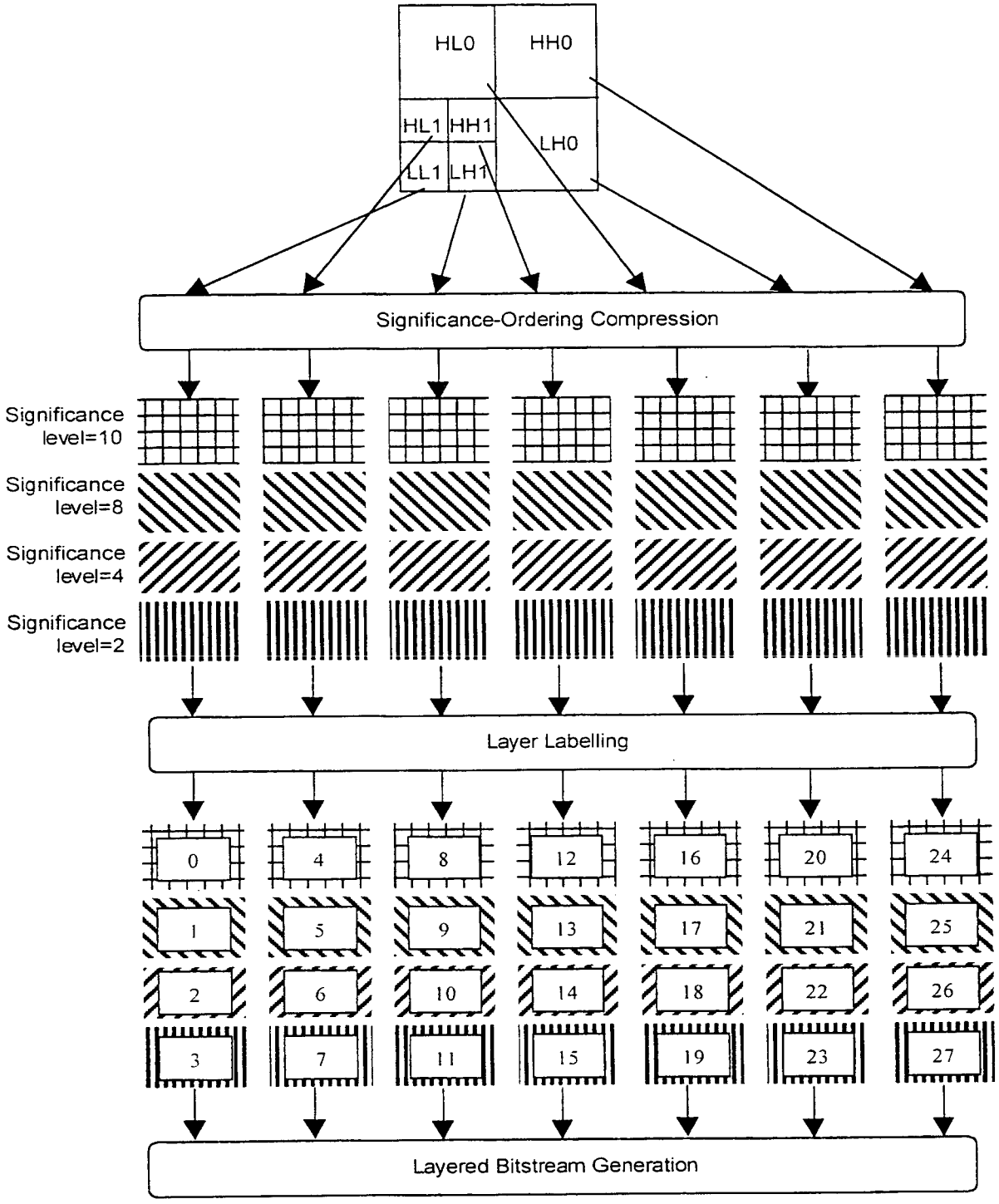


Figure 5

```
CodebookID: Wavelet_SIF_2D_YUV_0 ;
QualityTags: scale, fidelity, colour ;
QualityDivisions : scale(full, half, qtr), fidelity(lossless, high, medium, low), colour(luma, yuv) ;
OriginalSize: width=352, height=288 ;

Label: 0            // root subband, significan ce level 10
{
        Dependency: ;
        QualityParams : scale=qtr, fidelity=low, colour=luma ;
} ;

Label: 1            // root subband, significance level 8
{
        Dependency: 0 ;
        QualityParams : scale=qtr, fidelity=medium, colour=luma ;
} ;

Label: 2            // root subband, significance l evel 4
{
        Dependency: 1, 0 ;
        QualityParams : scale=qtr, fidelity=high, colour=luma ;
} ;

Label: 3            // root subband, significance level 2
{
        Dependency 2, 1, 0 ;
        QualityParams : scale=qtr, fidelity=lossless, colour=luma ;
} ;

Label: 4            // HL subband at depth  1, significance level 10
{
        Dependency: 3, 2, 1, 0 ;
        QualityParams : scale=half, fidelity=low, colour=luma ;
} ;

Label: 5            // HL subband at depth 1, significance level 10
{
        Dependency: 3, 2, 1, 0 ;
        QualityParams : scale=half, fidelity= medium, colour=luma ;
} ;

Label: 8            // LH subband at depth 1, significance level 10
{
        Dependency: 3, 2, 1, 0 ;
        QualityParams : scale=half, fidelity=low, colour=luma ;
} ;

Label: 9            // LH subband at depth 1, significance level 8
{
        Dependency: 3, 2, 1, 0 ;
        QualityParams : scale=half, fidelity=medium, colour=luma ;
} ;

Label: 12           // HH subband at depth 1, significance level 10
{
        Dependency: 8, 3, 2, 1, 0 ;
        QualityParams : scale=half, fidelity=low, colour=luma ;
} ;

Label: 13           // HH subband at depth 1, significance level 8
{
        Dependency: 12, 8, 3, 2, 1, 0 ;
        QualityParams : scale=half, fidelity=medium, colour=luma ;
} ;
```

Figure 6

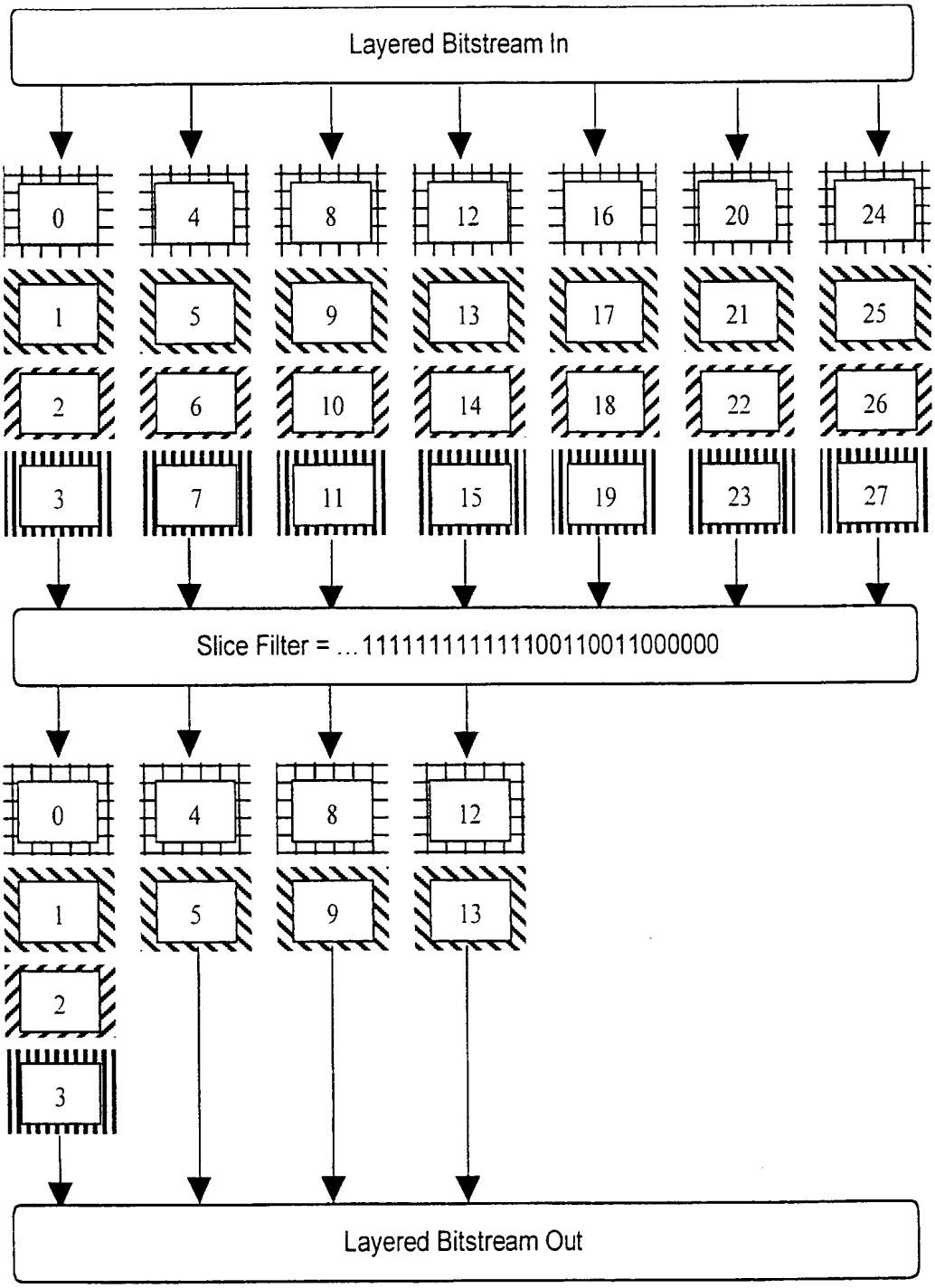


Figure 7

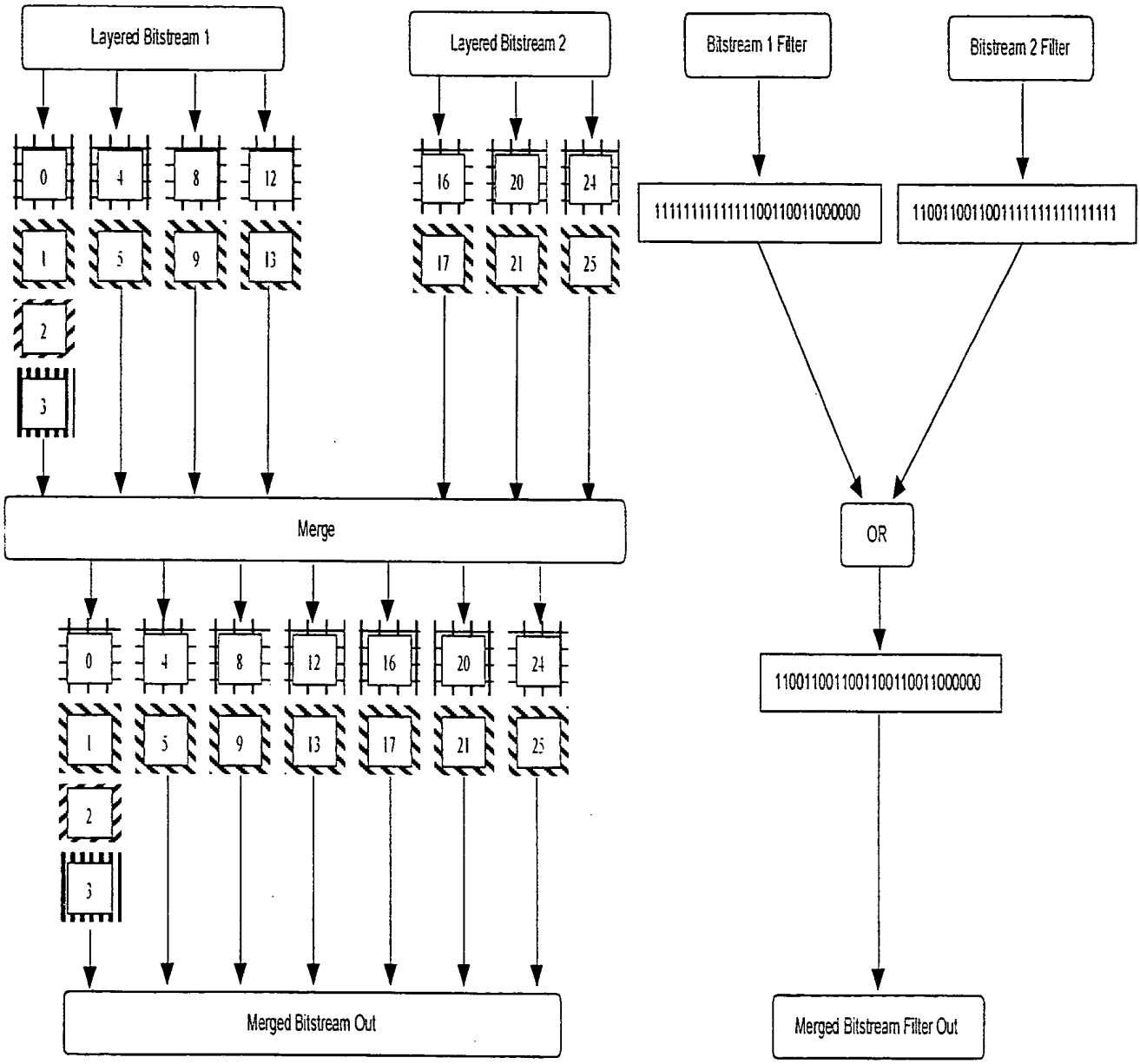


Figure 8

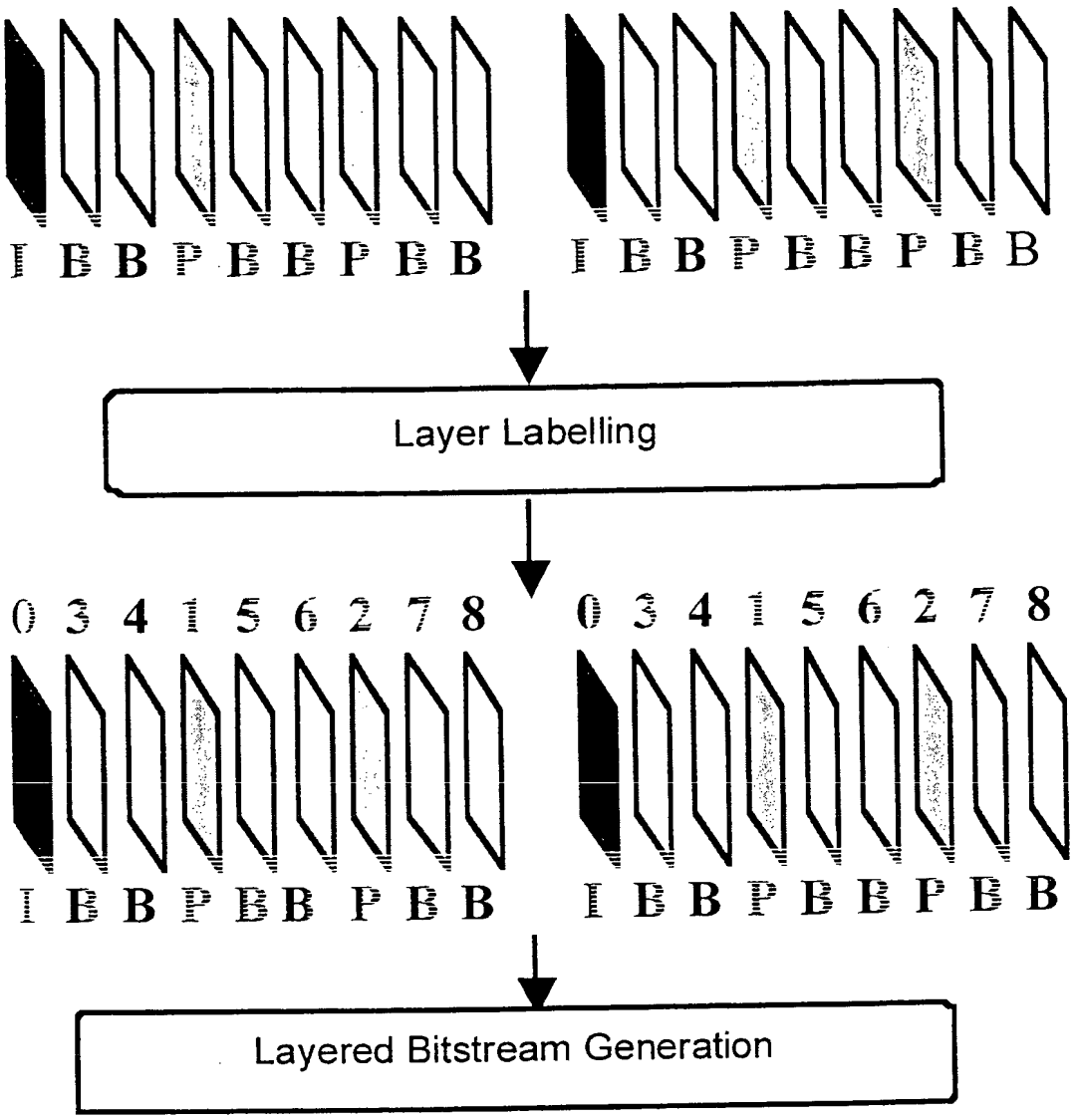


Figure 9

```
CodebookID: Mpeg1_GOP9 _0 ;
QualityTags : playrate ;
QualityDivisions : playrate (normal, fast, vfast) ;

Label: 0  // I-Frame
{
        Dependency : ;
        QualityParams : playrate=vfast ;
        TemporalPriority = 0 ;
} ;

Label: 1  // P-Frame
{
        Dependency : 0 ;
        QualityParams : playrate=fast ;
        TemporalPriority = 1 ;
} ;

Label: 3  // B-Frame
{
        Dependency : 1, 0 ;
        QualityParams : playrate=normal ;
        TemporalPriority = 2 ;
} ;
```

Figure 10

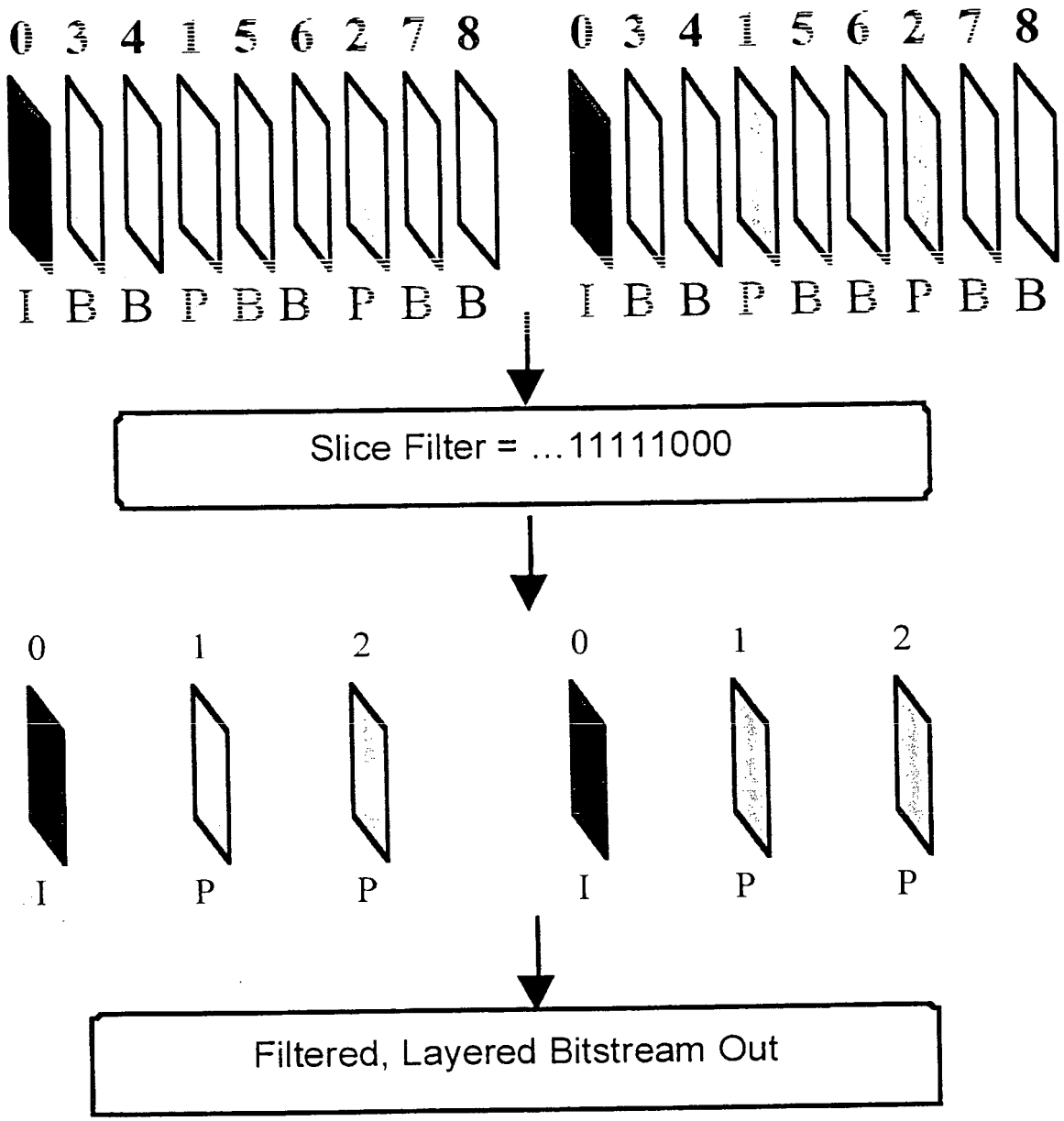


Figure 11

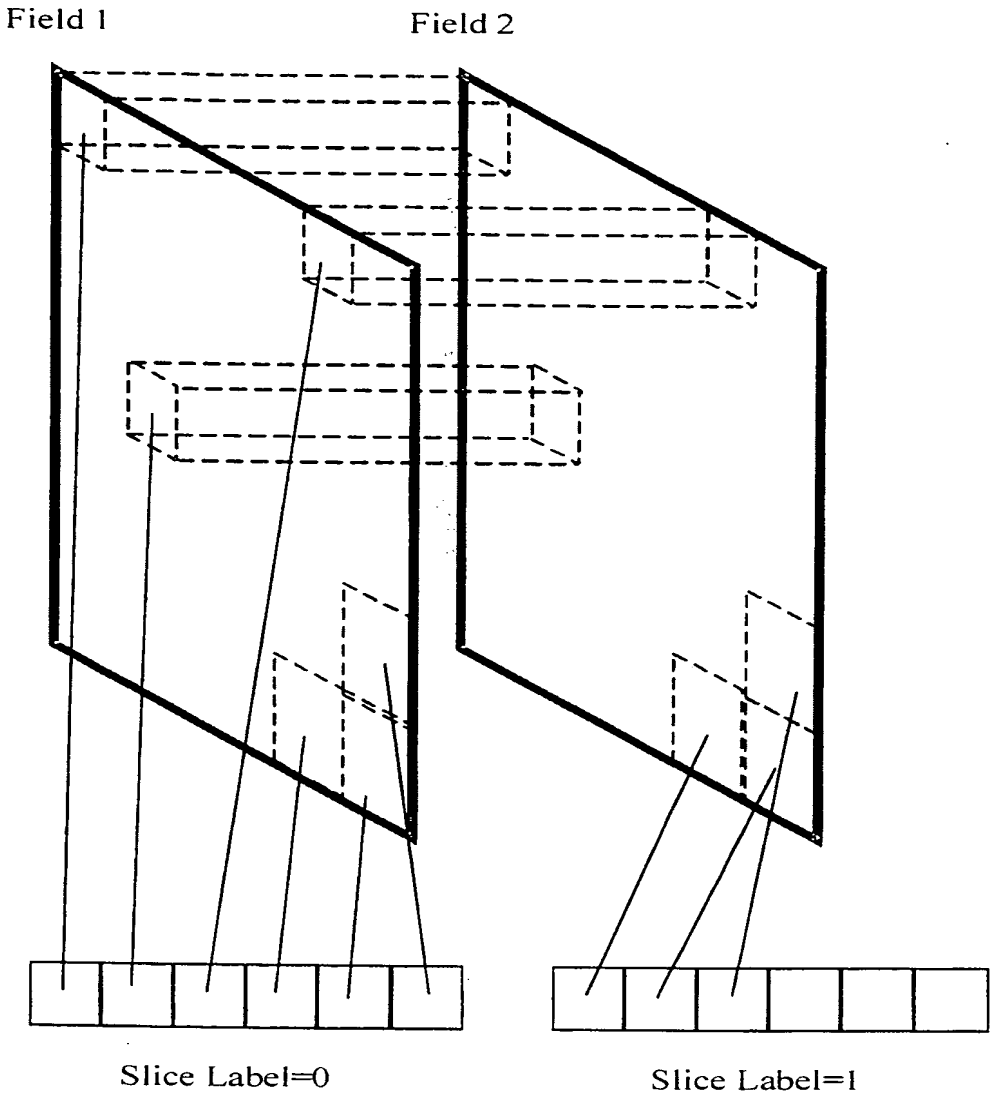


Figure 12

```
CodebookID: DV_0;
QualityTags: scale;
QualityDivisions: scale (full, half);
OriginalSize: width=704, height=576;

Label: 0  // Field 0 8-by-8 and field 0/1 2-by-4-by-8 DCT blocks
{
        Dependency: ;
        QualityParams: scale=half;
};

Label: 1  // Field 1 8-by-8 DCT blocks
{
        Dependency: 0;
        QualityParams: scale=full;
};
```

Figure 13

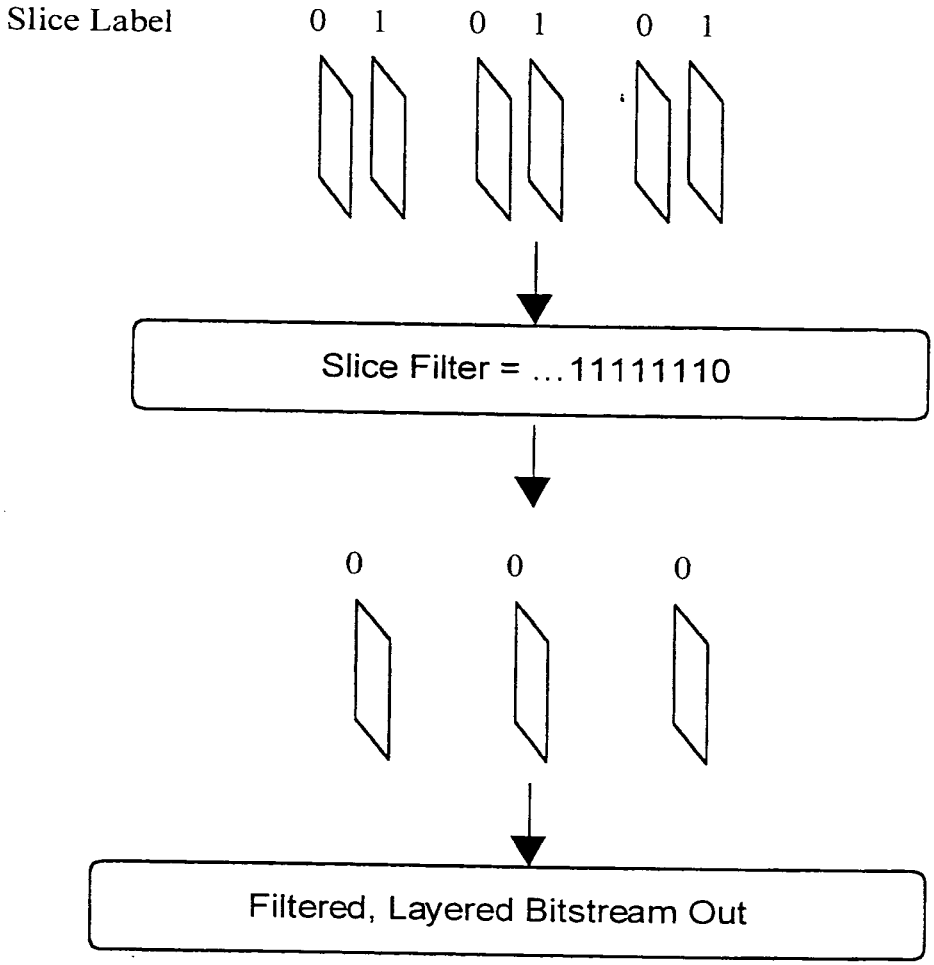

Slice Label
0 1 0 1 0 1
Slice Filter = ...11111110
0 0 0
Filtered, Layered Bitstream Out

Figure 14

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 November 2000 (02.11.2000)

PCT

(10) International Publication Number
WO 00/65838 A3

(51) International Patent Classification[7]: H04N 7/24, 7/26

(21) International Application Number: PCT/GB00/01614

(22) International Filing Date: 26 April 2000 (26.04.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
9909605.9 26 April 1999 (26.04.1999) GB

(71) Applicant *(for all designated States except US)*: **TELEMEDIA SYSTEMS LIMITED** [GB/GB]; Mount Pleasant House, 2 Mount Pleasant, Huntingdon Road, Cambridge CB3 0RN (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **KING, Tony, Richard** [GB/GB]; 28 Marlowe Road, Newnham, Cambridge CB3 9JW (GB). **GLAUERT, Timothy, Holroyd** [GB/GB]; 44 Whittlesford Road, Little Shelford, Cambridge CB2 5EW (GB). **WILSON, Ian, David** [GB/GB]; 26 The Limes, Harston, Cambridge CB2 5QT (GB). **EL WELL, Phil** [GB/GB]; 4 William Smith Close, Cambridge CB1 3QF (GB).

(74) Agent: **ORIGIN LIMITED**; 24 Kings Avenue, London N10 1PB (GB).

(81) Designated States *(national)*: GB, JP, US.

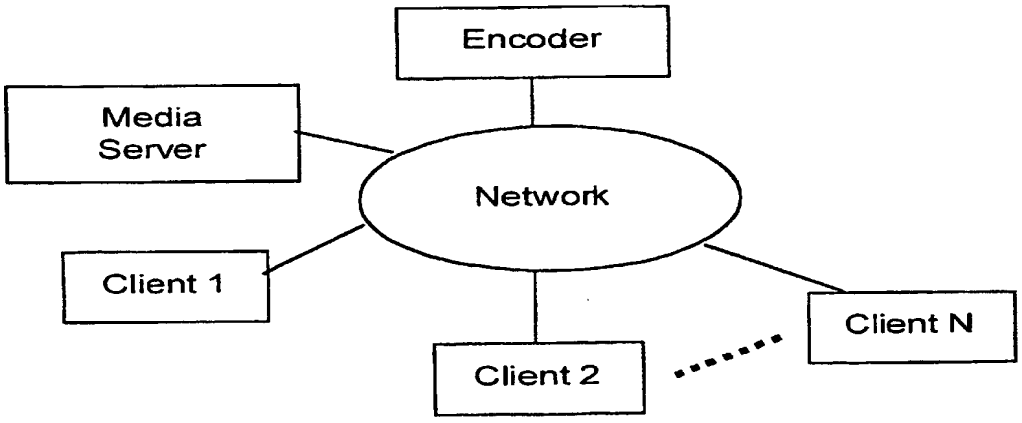
(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published:
— *with international search report*

(88) Date of publication of the international search report:
1 November 2001

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: CONVERSION OF A MEDIA FILE INTO A SCALABLE FORMAT FOR PROGRESSIVE TRANSMISSION



(57) Abstract: The delivery of a single encoded video or audio file which can be played back over a network to various clients at different data rates, resolutions and quality levels, as individually determined by each client, is disclosed. An encoder inserts into the media file bitstreams 'layer signalling' information which delimits and identifies a number of different layers (for example, different 'significance/scale layers' and different 'region layers'). A media Server stores the media files, and can distribute to different networked clients bitstreams with different properties depending upon the layers which are requested by or are appropriate to each client.

WO 00/65838 A3

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/GB 00/01614

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/24 H04N7/26

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 98 37699 A (INTERVU INC) 27 August 1998 (1998-08-27)	1-5,9, 11,12, 32,34-38
Y	abstract page 9, line 12 -page 10, line 4 page 11, line 31 -page 15, line 6 figure 3	7
X	WO 98 37698 A (ADAPTIVE MEDIA TECHNOLOGIES) 27 August 1998 (1998-08-27) cited in the application page 4, line 27 -page 5, line 2 page 6, line 23 -page 8, line 33	1,2,4,5, 9,11,12, 34-38

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
1 August 2000	09/08/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Hampson, F

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/GB 00/01614

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 96 29818 A (BHARATH ANIL ;KITNEY RICHARD (GB); IMPERIAL COLLEGE (GB)) 26 September 1996 (1996-09-26)	7
A	abstract page 2, line 25 -page 4, line 23	1
A	BEONG-JO KIM ET AL: "An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT)" PROCEEDINGS DCC '97. DATA COMPRESSION CONFERENCE (CAT. NO.97TB100108), PROCEEDINGS DCC '97. DATA COMPRESSION CONFERENCE, SNOWBIRD, UT, USA, 25-27 MARCH 1997, pages 251-260, XP002143648 1997, Los Alamitos, CA, USA, IEEE Comput. Soc. Press, USA ISBN: 0-8186-7761-9 abstract page 251, line 12 -page 252, line 18	1,7,8

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

.ormation on patent family members

Inter nal Application No
PCT/GB 00/01614

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9837699 A	27-08-1998	AU 6672298 A	09-09-1998
WO 9837698 A	27-08-1998	US 5953506 A	14-09-1999
		AU 5796598 A	09-09-1998
		EP 0945020 A	29-09-1999
WO 9629818 A	26-09-1996	AU 5012296 A	08-10-1996